CGS-808

Intelligent Color Graphics

Software Manual

Biotech Electronics

CGS-808

SOFTWARE MANUAL

September 1979

Biotech Electronics
P.O. Box 485
Ben Lomond, California 95005
(408) 338-2686

Copyright 1979
Printed in U.S.A.

TABLE OF CONTENTS

SECTION I		Page
1.0	FIRMWARE PACK I HARDWARE DESCRIPTION. .	1-1
1.1	EPROM Configuration	1-2
1.2	Status Code	1-2
1.3	I/O Ports and Handshaking Bits	1-3
SECTION 2		
2.0	FIRMWARE PACK I SOFTWARE DESCRIPTION. .	2-1
2.1	Passing Parameters	2-1
2.2	Memory Map	2-1
2.3	OP-Code Descriptions	2-1
2.4	8085 Interrupt Handling.	2-8
SECTION 3		
3.0	USER DEFINABLE OP-CODES	3-1
3.1	Writing Your Own OP-Code	3-1
3.2	Calling Another OP-Code	3-2
3.3	Helpful Hints	3-3
SECTION 4		
4.0	TESTING	4-1
4.1	Visual Inspection	4-1
4.2	Getting a Good Display	4-1
4.3	Demo Programs	4-2
4.3.1	BASIC Demo Program	4-2
4.3.2	Assembly Language Demo.	4-2
4.3.3	Color Bar Generator	4-5
4.3.4	Memory Test	4-6

LIST OF APPENDICES

Appendix	Description	Page
A	Firmware Pack I	A-1
B	OP-Code Format Summary	B-1
C	Detailed Description of VDG Modes	C-1

LIST OF FIGURES

Figure	Description	Page
1	Firmware Pack I OP-Codes	1-1
2	Clear Screen OP-Code	1-2
3	EPROM Configuration	1-2
4	Status Code	1-3
5	OP-Code Status	1-3
6	I/O Ports Used for Data Transfer	1-3
7	SW1 Address Compare Switch	1-4
8	Status Input Port Handshaking Status Bits	1-4
9	Firmware Pack I Flowchart	2-2
10	Initialization	2-4
11	Change Mode Subroutine	3-2
12	Clear Screen	3-3
13	OP-Code Parameters	3-3
14	Understanding OP-Code Formats	4-2
15	Demo CGS-808 Assembly Language Driver	4-4
16	BASIC Demo	4-5
17	Color Bar Generator	4-6
18	Memory Test	4-9

1.0 FIRMWARE PACK I HARDWARE DESCRIPTION

Firmware Pack I (FWP-I) is an alphanumeric and graphics driver software package for the CGS-808. FWP-I resides in a 2708 1K byte EPROM and has seven basic OP-Codes (See Figure 1).

OP-Code	Operation
0	Clear Screen
1	Set Graphics Mode
2	Plot Point
3	Draw Line
4	Alphanumeric/Semigraphic Driver
5	Read Screen or Data Base Memory
6	Write Screen or Data Base Memory

Figure 1 - Firmware Pack I OP-Codes

In addition to the seven OP-Codes, FWP-I takes care of transferring parameters between the S-100 bus and the on-board 8085. All data transferred to the CGS-808 is passed through two I/O ports on the S-100 bus.

The format for each OP-Code is basically the same --- output the OP-Code (0-6) to the port the dip switch (SW1) is set at; followed by outputting the necessary parameters to that same port address; then execute the OP-Code by issuing an output instruction to the dip switch address plus one. The status code is then tested to determine when the OP-Code is finished. The following example illustrates how the screen is cleared (See Figure 2).

```
0010 *
0020 * CLEAR SCREEN EXAMPLE
0030 * WRITTEN IN 8080 ASSEMBLY LANGUAGE
0040 *
0050 CODATA  EQU   00H          * ADDRESS OF DIP SWITCH
0060 STAT    EQU   01H          * ADDRESS OF DIP SWITCH + 1
0070 *
0080 BEGIN   EQU   $
0090         MVI   B,00         * CLEAR SCREEN OP-CODE: 0
0100         CALL  POUT         * PASS PARAMETER TO CGS-808
0110         MVI   B,80H        * CLEAR AND INITIALIZE
0120         CALL  POUT         * OUTPUT PARAMETER
0130         NOP                *
0140         NOP                *
0150         OUT   CODATA+1     * EXECUTE OP-CODE
0160 LOOP    IN    CODATA       * READ STATUS CODE
0170         CPI   00H          * TEST IF DONE WITH OP-CODE
0180         JNZ   LOOP         * NOT DONE?
0190         RET                * DONE
```

```
0200 POUT   EQU   $
0210        IN    STPT        * GET HARDWARE STATUS BITS
0220        RAR               * GET STATUS BIT 0
0230        JC    POUT        * LOOP TIL READY
0240        MOV   A,B         * GET DATA
0250        OUT   ODATA       * PASS DATA TO CGS-808
0260        RET               *
```

```
10 REM CLEAR SCREEN EXAMLPE
20 REM WRITTEN IN BASIC
30 P=0 \REM P=PORT ADDRESS
40 OUT P,0 \REM CLEAR SCREEN OP-CODE\ 0
50 OUT P,128 \REM CLEAR AND INITIALIZE
60 OUT P+1,0 \REM EXECUTE OP-CODE
70 S=INP(P) \REM READ STATUS CODE
80 IF S=192 THEN 90 ELSE 70 \REM TEST IF DONE
90 STOP \REM DONE
```

Figure 2 - Clear Screen OP-Code

1.1 EPROM Configuration

Firmware Pack I resides in a 2708 1K byte EPROM in socket U32. The first word address of U32 is location 0000. Jumpers W3 and W5 should be installed to connect the -5V and +12V supplies to U32 for the 2708 (Firmware Pack I). Another socket is provided for further expansion (U33). By changing the jumpers, both sockets (U32 and U33) can be configured for either 2708 or single 5V supply 2716 type EPROMs, allowing up to 4K bytes of ROM storage. Following are the four possible EPROM configurations (See Figure 3).

Socket U32
2708 Install jumpers W3 and W5*
2716 Install jumpers W4 and W6

Socket U33
2708 Install jumpers W8 and W10
2716 Install jumpers W7 and W9

*Firmware Pack I

Figure 3 - EPROM Configurations

1.2 Status Code

There are two types of status' for the CGS-808: 1) status code, and 2) status bits. The status code is data read from the I/O port which contains error information and status on the state of the software when executing the OP-Codes (See Figure 4).

Bit 0 NOT USED
1 Invalid OP-Code
2 Reset Required
3 NOT USED
4 NOT USED
5 NOT USED
6 Inputting Parameters (1 = Inputting Parameter)
7 Busy/Done Executing OP-Code
(1 = Done/0 = Busy)

Figure 4 - Status Code

Bits 1 and 2 indicate that an error condition occurred. If bit 1 is set, an invalid OP-Code was sent and the OP-Code has been terminated. Bit 2 indicates a hard error has occurred. The board should be reset to re-initialize the system parameters. Bits 6 and 7 are used to determine the state of the OP-Code (See Figure 5).

Bit 7 6
0 0 - Busy Executing OP-Code
0 1 - Inputting Parameters While Executing OP-Code
1 0 - Not Allowed
1 1 - Done Executing OP-Codes and Ready to Input Parameters

Figure 5 - OP-Code Status

1.3 I/O Ports and Handshaking Bits

The CGS-808 uses two parallel input and output ports to transfer data to and from the S-100 bus. The following I/O ports are used for data transfer for the CGS-808 (See Figure 6).

Output Port 0 - Data (OP-Code Parameters)
Output Port 1 - Initialize (Execute OP-Code)
Input Port 0 - Data or Status Code
Input Port 1 - Handshaking Status (Bits 0 and 1)

Figure 6 - I/O Ports Used for Data Transfer

The dip switch (SW1) can set the I/O port addresses to any four port boundary. The address decoder will decode four port addresses, but only two are used. SW1-Position 1 is address bit 2, or the least significant bit which is located toward the bottom of the board. SW1-Position 6 is address bit 7, or the most significant bit which is toward the top of the board. See Figure 7 for the possible port addresses.

SW1-1 = Bit A2 - LSB at Bottom of Board
SW1-6 = Bit A7 - MSB at Top of Board
Open = Off = 0; Closed = On = 1

SW1-6	On	On	On	Off	Off
SW1-5	On	On	On	On	On
SW1-4	On	On	On	On	Off
SW1-3	On	On	On	On	On
SW1-2	On	On	Off	On	On
SW1-1	On	Off	Off	On	On
Port	00H	04H	06H	80H	A0H

Figure 7 - SW1 Address Compare Switch

The status bits are two hardware handshaking bits used to determine the condition of the I/O ports.

The handshaking status input port 1 has the following bit definitions (See Figure 8).

Data Output from S-100 CPU
Bit 0 1 = Busy (Data Not Taken)
0 = Done (Ready for Next Byte)

Data Input to S-100 CPU
Bit 7 1 = No Data in Register
0 = Data Ready

Figure 8 - Status Input Port Handshaking Status Bits

2.0 FIRMWARE PACK I SOFTWARE DESCRIPTION

2.1 Passing Parameters

The heart of Firmware Pack I is the software that takes the parameters from the S-100 bus and passes them to the OP-Code subroutines (Refer to Appendix A). When the CGS-808 is powered on, the software will set the mode to the high density, clear the screen and initialize the system parameters. The program then enters the idle loop (lines 1950-2100) which continuously scans the S-100 I/O ports for OP-Code numbers and parameters. Once the necessary parameters have been sent to the CGS-808, the host processor must output to the port address plus one to execute the OP-Code (lines 1680-1840). This generates an interrupt causing the program to call the DRVR subroutine. The DRVR subroutine (lines 2390-2850) calculates the OP-Code address and jumps to the OP-Code. When the OP-Code is finished the program goes back to the idle loop (See Figure 9).

2.2 Memory Map

The CGS-808 has 1K bytes of RAM storage for parameter or data bases. The beginning of the 1K RAM is located at 4000H. Firmware Pack I uses the first 256 bytes of this RAM for parameter storage. This block is divided up into several sections:

PBUFF 4000H-402FH - This is used for system parameters, constants for the OP-Code, and temporary storage for some variables the OP-Codes use.

PRAM 4030H-40CFH - This area is used for storing parameters entered into the CGS-808 from the S-100 bus. Location 4030H is the OP-Code number, 4031 is the first parameter, 4032 the second parameter, etc.

STACK 40FFH-40CFH - This is the top of the stack (30 bytes).

DBASE 4100H-43FFH - This is a data base area of 768 bytes. This is not used in FWP-1, but is used in FWP-2 for storing custom character sets.

2.3 OP-Code Descriptions

OP-Code	Description
0	Clear Screen - The clear screen subroutine will clear

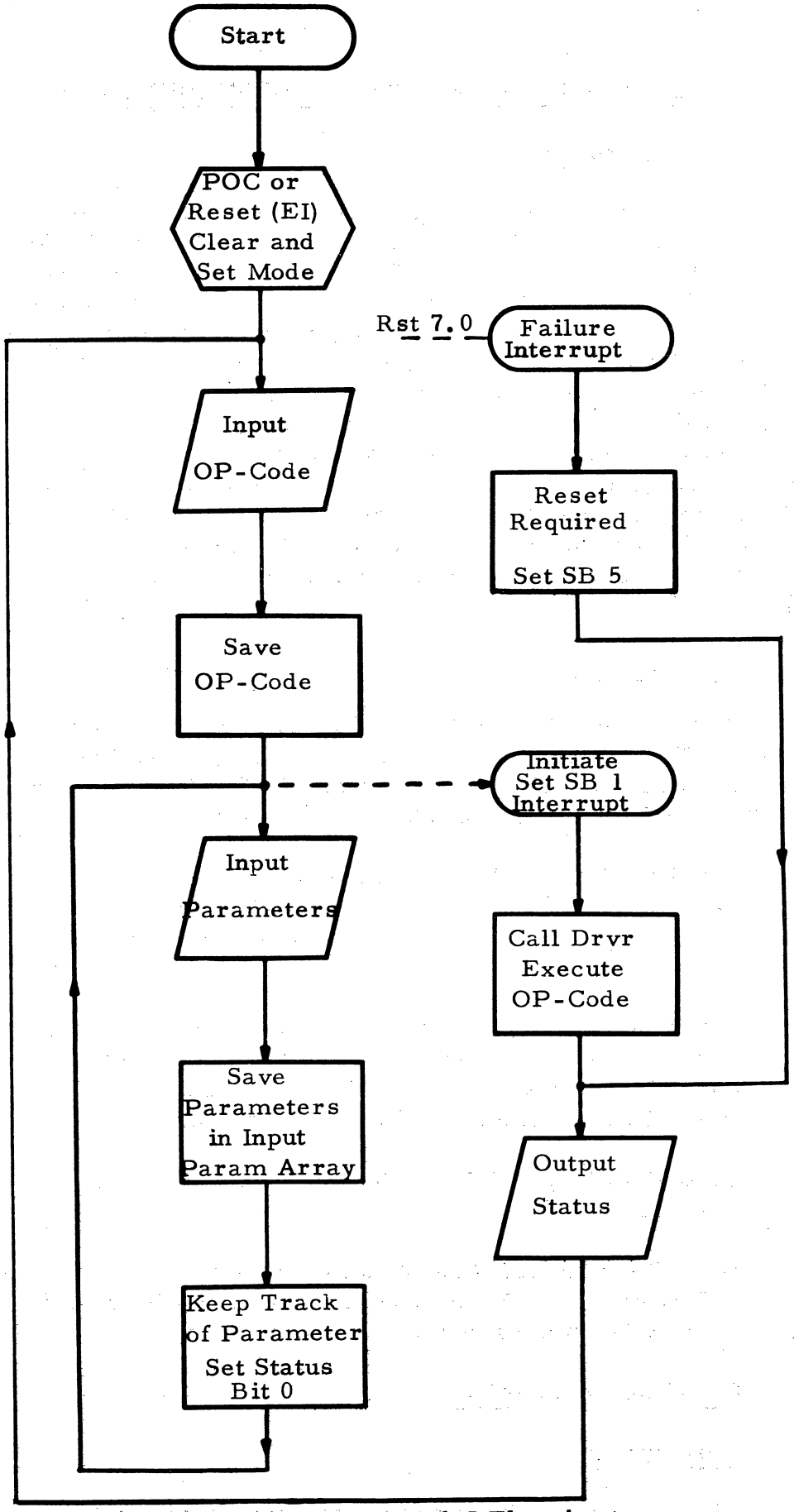

Figure 9 - Firmware Pack I Flowchart

OP-Code Description

all 6,144 bytes of the screen refresh memory to any color. The clear screen can also fill the half word memory when in the alphanumerics mode. If bit 7 is set, certain system parameters will be initialized. The clear screen uses one parameter --- Z.

Parameter Bit Definition:

7	6	5	4	3	2	1	0
INT	X	X	CSS	MD1	MD2	CZ0	CZ1

CZ0 and CZ1 determine the intensity or background color the screen will be cleared to.

Two Color Mode (CZ1 = Don't Care)

CZ0	CSS = 0	CSS = 1
0	Black	Black
1	Green	Buff

Four Color Mode

CZ1	CZ0	CSS = 0	CSS = 1
0	0	Green	Buff
0	1	Yellow	Cyan
1	0	Blue	Magenta
1	1	Red	Orange

MD0 and MD1 determine the mode the half-word memory will be set to and correspond to $\overline{INT}$/EXT and $\overline{A}$/S of the MC6847 VDG. These bits are valid only when the mode has been previously set to alphanumerics.

MD1	MD0	Mode
0	0	Alphanumerics
0	1	NOT USED
1	0	Semigraphics-4
1	1	Semigraphics-6

CSS is used when in the alphanumeric/semigraphics mode to determine the character color or color set for semigraphics-6. See Table 3 of Appendix C for a detailed description.

INT, if set, will initialize the system parameters. When the board is powered-up, the system parameters will be initialized and the screen refresh memory is cleared.

OP-Code Description

Figure 10 shows exactly which parameters are initialized when bit 7 (INT) is set.

```
014F 3A 03 40          3900 SINL  LDA   Z          * GET Z
0152 1F                3910       RAL             * TEST IF INITIALIZE BIT SET
0153 D2 B2 00          3920       JNC   CRTX      * RETURN
0156 AF                3930       XRA   A         * CLEAR ACC
0157 32 1C 40          3940       STA   SHORT+1   *
015A 32 1F 40          3950       STA   FLAG1     * USER FLAG CLEARED
015D 32 20 40          3960       STA   FLAG2     * USER FLAG CLEARED
0160 32 1D 40          3970       STA   LOCUR     * CLEAR CURSOR POSITION
0163 32 03 40          3980       STA   THREED    * THREE D MODE
0166 D3 03             3990       OUT   03H       * CLEAR INTERRUPTS
0168 3E 60             4000       MVI   A,60H     * INITIALIZE SCREEN ADDR
016A 32 1E 40          4010       STA   LOCUR+1   *
016D 3E 80             4020       MVI   A,80H     * DONE OP-CODE STARTS
016F 32 07 40          4030       STA   STAT      * CLEAR STATUS
0172 3E 03             4040       MVI   A,03H     * SET UPDATE X, UPDATE Y MODE
0174 32 04 40          4050       STA   UPMOD     *
0177 C3 B2 00          4060       JMP   CRTX      * RETURN
```

Figure 10 - Initialization

1 Set Mode - The set mode subroutine will set the graphics mode register (U39). The set mode subroutine also determines boundary conditions for the plot point and draw line subroutine and should be executed prior to using these subroutines. The set mode uses one parameter --- GM.

Parameter Bit Definition:

7	6	5	4	3	2	1	0
0	0	EXT	CSS	$\overline{A}$/G	GM2	GM1	GM0

GM0, GM1 and GM2 control the graphics mode when $\overline{A}$/G is high. See Table 3 of Appendix C for a detailed description.

$\overline{A}$/G, when high, will enable the full graphics mode; when low will select the alphanumeric/semigraphics mode.

CSS, when in the full graphics mode ($\overline{A}$/G) high), will select the color set. When in the alphanumerics mode, this signal is not used.

OP-Code	Description

CSS = 0	CSS = 1
Green	Buff
Yellow	Cyan
Blue	Magenta
Red	Orange

EXT, when high, will actuate the external video signal (used with the RBY Adapter board).

Bits 6 and 7 should always be zero.

2 Plot Point - The mode must be set prior to execution of the plot point OP-Code. The origin (X = 0, Y = 0) is at the lower left hand corner. The X, Y coordinates are positive values in the first quadrant. The plot point subroutine uses three parameters --- X, Y and Z.

Parameter Definitions:

X X Coordinate

Y Y Coordinate

Z Color or intensity (depending on the mode)

Two Color Mode

	Bit 0	CSS = 0	CSS = 1
Z =	0	Black	Black
	1	Green	Buff

Four Color Mode

	Bit 1	Bit 0	CSS = 0	CSS = 1
Z =	0	0	Green	Buff
	0	1	Yellow	Cyan
	1	0	Blue	Magenta
	1	1	Red	Orange

3 Draw Line - There are no restrictions for the direction the lines may be drawn. The draw line subroutine uses five parameters --- X_S, Y_S, X_E, Y_E, and Z.

X_S	Starting X Coordinate
Y_S	Starting Y Coordinate
X_E	Ending X Coordinate
Y_E	Ending Y Coordinate
Z	Color Intensity (Same as Plot Point Z)

OP-Code	Description
4	Alphanumeric/Semigraphic - The alphanumeric/semigraphics subroutine can display both alphanumerics and semigraphics data on a character by character basis. Carriage returns and back spaces are recognized in the alphanumeric mode. Full cursor control is provided by specifying the character position and line position. Four parameters are used --- data, character mode, character position and line position.

Parameter Definition:

1. Data - Depending on the mode, there are three possible formats:

Alphanumerics

7	6	5	4	3	2	1	0
Inv Video	X	ASCII					

Semigraphics-4

7	6	5	4	3	2	1	0
X	C2	C1	C0	L3	L2	L1	L0

Semigraphics-6

7	6	5	4	3	2	1	0
C1	C0	L5	L4	L3	L2	L1	L0

C2 is the CSS bit in the character mode parameter and is used to obtain eight colors in semigraphics-6. Refer to Table 3 in Appendix C.

2. Character Mode - The character mode programs the half word data on a character by character basis.

Bit Definition:

7	6	5	4	3	2	1	0
X	X	X	X	CSS	MD1	MD0	X

OP-Code Description

MD0	MD1	Mode
0	0	Alphanumerics
0	1	NOT USED
1	0	Semigraphics-4
1	1	Semigraphics-6

The color select signal (CSS) determines the character color, or is C2 in the semigraphics-6 mode.

3. Character Position - Defines the position the cursor will be from the left hand margin (ranges from 0 to 31).

4. Line Position - Defines the line the cursor will be on (ranges from 0-15). When the CGS-808 is powered on or initialized, the cursor will be initialized to the upper left hand corner of the screen (i.e. character position equals zero, line position equals zero).

5 Read Screen - The read screen subroutine can transfer any part of the CGS-808 memory to the host computer. One (1) to 255 bytes may be transferred at a time. The read screen subroutine uses four parameters.

Parameter Bit Definition:

1. Lower eight bits of memory address -

7	6	5	4	3	2	1	0
A7	A6	A5	A4	A3	A2	A1	A0

2. Upper five bits of memory address -

7	6	5	4	3	2	1	0
0	1	1	A12	A11	A10	A9	A8

Bits 7, 6 and 5 are set up to read the screen refresh memory.

3. Mode - Bits 6 and 7 are used to enable either the half word or full word memory. Bits 6 and 7 should never be enabled at the same time. 80H will enable the full word; 40H will enable the half word.

OP-Code Description

7	6	5	4	3	2	1	0
FW	HW	X	X	X	X	X	X

4. Number of bytes transferred (N) - The number of bytes transferred can range from 1 to 255. The bytes are then inputted to the host CPU, from 1 to N, by inputting port 0.

6 Write Screen - The write screen subroutine will transfer, from the host system, any part of the CGS-808 memory. One (1) to 255 bytes may be transferred at a time. The write screen subroutine uses four parameters. The four parameters are the same as the parameters used in the read screen OP-Code (OP-Code 5). The bytes are outputted, from 1 to N, to port 0.

Refer to Appendix B for a summary of the OP-Codes in Firmware Pack I.

2.4 8085 Interrupt Handling

There are three interrupts the 8085 uses on the CGS-808: RST 7.5, 6.5 and 5.5. Firmware Pack I uses only RST 7.5, and masks out RST 6.5 and RST 5.5. RST 7.5 is used to exit the idle loop. The idle loop constantly scans the S-100 I/O ports for new OP-Codes or more parameters. When the user outputs any data to the port address plus one, an RST 7.5 interrupt is generated causing the program to jump to location 3CH.

To use the expansion port, RST 6.5 and RST 5.5 must be enabled by a SIM instruction (30H), with the mask bits loaded in the accumulator (18H) prior to executing the SIM instruction. When data has been taken from the expansion output port (U10), the hardware generates a RST 5.5 causing the processor to vector to location 2003H. When data has been clocked into the input expansion port (U9) the processor will vector to location 2006H. After either interrupt is serviced, the interrupt response flip-flops (U21) must be cleared by an output instruction to port 03.

3.0 USER DEFINABLE OP-CODES

3.1 Writing Your Own OP-Code

User definable OP-Codes can easily be added by using the second EPROM socket (U33) at location 2000H.

Any OP-Code which is greater than six but less than thirty will cause the DRVR subroutine (lines 2390-2850 of Appendix A) to search for the OP-Code addresses in a jump table at 200AH instead of at line 2750. The addresses in the jump table are Intel byte-swapped; for example, the beginning address for OP-Code 7 would be located at 200AH (high address byte) and 200BH (low address byte). The address for OP-Code 8 would follow.

When the DRVR subroutine jumps to the OP-Code, the accumulator contains the number of parameters, and the D,E registers contain the first address of the parameters buffer. An LDAX D instruction will load the accumulator with the first parameter. An INX D, LDAX D instruction will load the next parameter, etc. To exit the OP-Code, a JMP CRTX (location 00B2H) must be the last instruction of the OP-Code.

The following is an example of an OP-Code routine (See Figure 11).

```
00B7                    2860 *
00B7                    2870 *
00B7                    2880 *
00B7                    2890 * CHANGE MODE ROUTINE
00B7                    2900 *
00B7                    2910 SMODE  EQU   $
00B7 1A                 2920        LDAX  D          * GET MODE
00B8 F6 80              2930        ORI   80H        * ENABLE FW MEM
00BA 32 05 40           2940        STA   MD         * SAVE MODE
00BD D3 01              2950        OUT   MDPT       * CHANGE MODE
00BF 47                 2960        MOV   B,A        *
00C0 E6 08              2970        ANI   08H        * CHCEK IF IN GRAPHICS MODE
00C2 CA B2 00           2980        JZ    CRTX       * RET IF ANOT/G IS 0
00C5 21 DC 00           2990        LXI   H,PLTBL    * TABLE FOR NOL,BPL
00C8 78                 3000        MOV   A,B        * GET MODE
00C9 E6 07              3010        ANI   07H        * GET GRAPHICS MD=ADDRESS
00CB 07                 3020        RLC              * MULTIPLY BY 2
00CC 4F                 3030        MOV   C,A        * BC= BIAS ADDRESS
00CD 06 00              3040        MVI   B,00H      * CLEAR UPPER BITS
00CF 09                 3050        DAD   B          * ADD DISPLACEMENT
00D0 7E                 3060        MOV   A,M        * GET NOL
00D1 32 00 40           3070        STA   NOL        * SAVE NOL
00D4 23                 3080        INX   H          * GET BPL ADDRESS
00D5 7E                 3090        MOV   A,M        * GET BPL
```

```
00D6 32 01 40          3100        STA   BRL          * SAVE BPL
00D9 C3 B2 00          3110        JMP   CRTX         * RETURN
00DC                   3120 *
00DC                   3130 * PLOT TABLE FOR NOL & BPL
00DC                   3140 *
00DC                   3150 PLTBL  EQU   $
00DC 3F                3160        DB    63
00DD 10                3170        DB    16
00DE 3F                3180        DB    63
00DF 10                3190        DB    16
00E0 3F                3200        DB    63
00E1 20                3210        DB    32
00E2 5F                3220        DB    95
00E3 10                3230        DB    16
00E4 5F                3240        DB    95
00E5 20                3250        DB    32
00E6 BF                3260        DB    191
00E7 10                3270        DB    16
00E8 BF                3280        DB    191
00E9 20                3290        DB    32
00EA BF                3300        DB    191
00EB 20                3310        DB    32
00EC                   3320 *
```

Figure 11 - Change Mode Routine

3.2 Calling Another OP-Code

In some cases, it may be necessary to have an OP-Code call another OP-Code. The best way to accomplish this is to first fill the beginning of parameter storage plus one (PRAM plus one or location 4031H) with the parameters to be passed to the OP-Code. After all the parameters have been entered (PRAM plus one, PRAM plus two, etc.), initialize the H, L register pair to PRAM plus one (LXI H, 4031H), load the accumulator with the OP-Code number and CALL DRVR (Location 0084H). This procedure is similar to lines 1020-1120 of Appendix A. The following is an example of how the screen is cleared calling an OP-Code (See Figure 12).

```
0010 *
0020 * EXAMPLE- ONE OP-CODE CALLING ANOTHER
0030 *
0040 * ROUTINE TO SET THE MODE AND CLEAR SCREEN
0050 * OP-CODE 7 - NO PARAMETERS
0060 *
0070 EXAMP  EQU   $
0080        LXI   H,PRAM+1        * BEGIN OF PARAMETER STORAGE
0090        MVI   A,01            * OP-CODE 1
0100        MVI   M,8FH           * HIGH DENSITY MODE
0110        CALL  DRVR            * EXECUTE OP-CODE
0120        XRA   A               * OP-CODE 0
0130        MVI   M,80H           * H,L STILL POINT TO PRAM+1
```

```
0140        CALL   DRVR          * CLEAR AND INITIALIZE
0150        JMP    CRTX          * RETURN TO IDLP
```

Figure 12 - Clear Screen

3.3 Helpful Hints

1. When the DRVR subroutine (lines 2390-2850) jumps to the OP-Code, the accumulator contains the number of parameters which can be used by the OP-Code to to fetch from the parameter storage (PRAM plus one) the correct number of parameters.

2. It is not always necessary to specify all the parameters to the OP-Code. Once the parameter has been entered into PRAM plus one, they remain there until changed. For example, after once specifying X, Y and Z for the plot point OP-Code, only X and Y are needed the next time if Z does not change (See Figure 13).

   Output OP-Code
   Output X
   Output Y
   Output Z
   Output Initiate
   Next -
   Output OP-Code
   Output X
   Output Initiate

   Figure 13 - OP-Code Parameters

3. The clear screen is capable of initializing certain parameters when bit 7 is set in the clear screen parameter. The initialize routine will put zeros at location 401FH (FLAG 1) and 4020H (FLAG 2). These can be used by the user to have parameters cleared during power-up or clear and initialize OP-Code.

4. The set mode routine does more than just set the mode. It also calculates the number of lines and bytes per line for the plot point or draw line subroutine. Any user OP-Codes which use the plot point or draw line should first set the mode using the set mode OP-Code, and not by just setting to the mode register (U39).

5. DBASE (location 4100H) is the beginning of the data base which is 768 bytes long. This area is free to

the user. Firmware Pack II uses this area for character storage.

6. The expansion ports both jump to the second EPROM socket (U33). The input expansion port will jump to location 2006H. The output expansion port will jump to 2003H.

4.0 TESTING

The CGS-808 is a sophisticated board. A diagnostic program is provided to test the CGS-808. The program will test all of the screen refresh memory and the I/O ports, plus most of the Firmware Pack I software.

4.1 Visual Inspection

The first step that should be taken after all the components are stuffed on the PC board, is to give the board a thorough visual inspection. The board should be cleaned by using a stiff brush to remove small conductive particles. Next, examine the board for solder bridges, components not correctly oriented, anything that looks incorrect. On the component side, examine each IC for pins that are turned under. They are not easy to spot, but bent pins can cause considerable loss of time, money and energy. Look at each pin very closely. On the solder side, look for joints which are not shiny (cold solder joints). If you find any, re-solder them.

4.2 Getting a Good Display

To interface the CGS-808 to a commercial color T. V., a modulator is required if the T. V. does not allow for a direct video input.

When you apply power, the screen should be cleared and set at the highest density graphics mode in green. If the display is random graphics, then reset the board (with the reset jumper installed). The display should then be cleared. If the display is not in color, there are two possibilities: 1) The T. V. is not tuned in properly, 2) The trimmer capacitor (C4) is not adjusted properly. The board needs no further adjustments. Adjusting the tint, brightness and contrast on your T. V. will produce a brilliant color display.

The following diagnostic programs will test the screen refresh memory and aid in the adjustment of the color. Two programs are provided: 1) memory test program, 2) color bar generator program which will display the eight colors the board is capable of producing. The memory test program is written in 8080 assembly language, while the color bar generator is written in BASIC. The colors generated are: green, yellow, blue, red, buff, cyan, magenta and orange. The additional demo programs provided will help in understanding the OP-Code formats.

4.3 Demo Programs

4.3.1 BASIC Demo Program

The following program is written entirely in BASIC. It will help in understanding the OP-Code formats. The only requirements of the BASIC is that it must be able to INPUT or OUTPUT to I/O ports.

The program will ask for an OP-Code, then print the status code. The program will then enter a loop of inputting parameters. To get out of this loop (all necessary parameters have been entered), just hit return and the OP-Code will be executed. This program does not test any of the handshaking bits (See Figure 14).

```
10 REM DEMO PROGRAM TO HELP IN UNDERSTANDING
20 REM OP-CODE FORAMTS.
30 M=128 'REM ADDRESS SWITCH SETTING
40 INPUT "OP-CODE: ",C
50 Z=INP(M) 'REM INPUT STATUS
60 PRINT "STATUS= ",Z
70 OUT M,C 'REM OUTPUT OP-CODE
80 REM INPUT PARAMETERS
90 REM TO EXECUTE THE OP-CODE JUST HIT RETURN
100 INPUT "PARAMETER: ",P$
110 IF P$=""THEN 170
120 P=VAL(P$)
130 OUT M,P
140 Z=INP(M)
150 PRINT "STATUS= ",Z
160 GOTO 100
170 OUT M+1,0
180 Z=INP(M)
190 PRINT "STATUS= ",Z
200 GOTO 40
```

Figure 14 - Understanding OP-Code Formats

4.3.2 Assembly Language Demo

The following assembly language subroutine can be used to interface the CGS-808 to a BASIC program or another assembly language program. The OP-Code and parameters are passed to the subroutine by a parameter table (PTBL). Once the parameters are entered into the table, the subroutine can be called. The subroutine will input the parameters and execute the OP-Code, then return to the main program (See Figure 15). The assembly language program must be loaded into memory prior to running the BASIC program (See Figure 16).

```
0000                    0010 *
0000                    0020 * COPYRIGHT BIOTECH ELECTRONICS 1979
0000                    0030 * ALL RIGHTS RESERVED
0000                    0040 *
0000                    0050 * PROGRAMMER: RICK STELLMACHER
0000                    0060 * SOURCE FILE: ADEMO
0000                    0070 * VERSION: 1.0
0000                    0080 *
0000                    0090 * ADEMO- CGS-808 ASSEMBLY LANGUAGE DRIVER
0000                    0100 *
0000                    0110 * AFTER FILLING THE PARAMETER TABLE THEN
0000                    0120 * CALL ADEMO TO EXECUTE THE OP-CODE.
0000                    0130 *
0000                    0140 *
0000                    0150 * EQUATES
0000                    0160 *
0000                    0170 IDATA  EQU   80H          * INPUT PORT DATA
0000                    0180 ODATA  EQU   80H          * OUTPUT PORT DATA
0000                    0190 STPT   EQU   81H          * STATUS BITS- 0,7
0000                    0200 INIT   EQU   81H          * INITIATE PORT
0000                    0210 *
0000                    0220 *
0000                    0230 ADEMO  EQU   $            *
0000 21 38 00           0240        LXI   H,PTBL       * PARAMETER TABLE
0003 DB 80              0250        IN    IDATA        * GET STATUS
0005 77                 0260        MOV   M,A          * SAVE STATUS
0006 FE C0              0270        CPI   192          * ERROR?
0008 C2 23 00           0280        JNZ   ERR          *
000B 23                 0290        INX   H            * BUMP POINTER
000C 46                 0300        MOV   B,M          * GET OP-CODE
000D CD 24 00           0310        CALL  POUT         * SEND OP-CODE TO CGS-808
0010 23                 0320        INX   H            * NEXT PARAMETER
0011 56                 0330        MOV   D,M          * GET NO. OF PARAMETERS
0012 23                 0340 PLP    INX   H            * NEXT PARAMETER
0013 46                 0350        MOV   B,M          * GET PARAMETER
0014 CD 24 00           0360        CALL  POUT         *
0017 15                 0370        DCR   D            * COUNT DOWN PARAMETERS
0018 C2 12 00           0380        JNZ   PLP          * NOT DONE?
001B 3E 20              0390        MVI   A,20H        * DELAY FOR MORE THAN 100US
001D 3D                 0400 LOOP   DCR   A            *
001E C2 1D 00           0410        JNZ   LOOP         *
0021 D3 81              0420        OUT   INIT         * EXECUTE OP-CODE
0023 C9                 0430 ERR    RET                *
0024                    0440 *
0024                    0450 * THIS ROUTINE SEND DATA TO THE CGS-808
0024                    0460 *
0024                    0470 POUT   EQU   $            *
0024 DB 81              0480        IN    STPT         * GET HANDSHAKING BITS
0026 1F                 0490        RAR                * BIT 0
0027 DA 24 00           0500        JC    POUT         * IS REG EMPTY?
002A 78                 0510        MOV   A,B          * YES
002B D3 80              0520        OUT   ODATA        * OUTPUT PARAMETER
002D C9                 0530        RET
002E                    0540 *
```

```
002E                    0550 * THIS ROUTINE GET DATA FROM THE CGS-808
002E                    0560 *
002E                    0570 PIN    EQU  $              *
002E DB 01              0580        IN   STPT           * GET HANDSHAKING BITS
0030 17                 0590        RAL                 * BIT 7
0031 DA 2E 00           0600        JC   PIN            * IS REG FULL?
0034 DB 00              0610        IN   IDATA          * YES, GET DATA
0036 47                 0620        MOV  B,A            * RET DATA IN B REG
0037 C9                 0630        RET                 *
0038                    0640 *
0038                    0650 * THIS IS THE START OF THE PARAMETER TABLE
0038                    0660 *
0038                    0670 PTBL   EQU  $
0038 00                 0680        DB   00             * STATUS
0039 00                 0690        DB   00             * OP-CODE
003A 00                 0700        DB   00             * NO. OF PARAMETERS
003B                    0710 * THE REST IS RESERVED FOR PARAMETERS 1 THRU N
003B                    0720 *

SYMBOL TABLE

ADDR 0000   ERR  0023   IDATA 0000   INIT 0001   LOOP 001D
ODATA 0000  PIN  002E   PLP  0012   POUT 0024   PTBL 0038
STPT 0001

D 0000 0037
0000 21 38 00 DB 00 77 FE C0 C2 23 00 23 46 CD 24 00
0010 23 56 23 46 CD 24 00 15 C2 12 00 3E 20 3D C2 1D
0020 00 D3 01 C9 DB 01 1F DA 24 00 78 D3 00 C9 DB 01
0030 17 DA 2E 00 DB 00 47 C9
```

Figure 15 - Demo CGS-808 Assembly Language Driver

```
10 REM - PROGRAM TO INPUT PARAMETERS
20 REM THE ASSEMBLY LANGUAGE PROGRAM MUST BE
30 REM ENTERED BEFORE RUNNING THIS PROGRAM.
40 M=128 \ REM PORT 80-HEX
50 A=56 \ REM STATUS BYTE IN PARAMETER TABLE
60 Z=EXAM(A)
70 IF Z=192 THEN 130
80 !"ERROR: ",Z
90 REM CLEAR AND INITIALIZE CGS-808
100 OUT M,0
110 OUT M,128
120 OUT M+1,0
130 INPUT "OP-CODE= ",C
140 FILL A+1,C
150 INPUT "NO. OF PARAMETERS= ",N
160 FILL A+2,N
170 FOR I=1 TO N
180 INPUT "PARAMETER= ",P
```

```
190 FILL A+2+I,P
200 NEXT I
210 REM PARAMETER TABLE FILLED
220 S=CALL(0)
230 GOTO 50
```

Figure 16 - BASIC Demo

4.3.3 Color Bar Generator

The following BASIC program will display the eight colors the CGS-808 is capable of generating. The colors generated are: green, yellow, blue, red, buff, cyan, magenta and orange (See Figure 17).

```
10 REM COLOR BAR GENERATOR
20 P=128 \ REM PORT ADDRESS
30 D=6 \ REM HALF WORD DATA
40 M=64 \ REM MODE=64
50 S1=0 \ S2=96 \ REM SCREEN ADDRESS
60 GOSUB 1000
70 S1=128
80 GOSUB 1000
90 S1=0 \ S2=97
100 D=14
110 GOSUB 1000
120 S1=128
130 GOSUB 1000
140 D=63 \ REM FULL WORD DATA
150 M=128 \ REM MODE=128
160 S1=0 \S2=96 \ REM SCREEN ADDRESS
170 GOSUB 1000
180 S1=128
190 GOSUB 1000
200 S1=0\S2=97
210 D=63
220 GOSUB 1000
230 S1=128
240 GOSUB 1000
250 STOP
1000 REM WRITE SCREEN OP-CODE 6
1010 OUT P,6
1020 OUT P,S1
1030 OUT P,S2
1040 OUT P,M
1050 OUT P,128
1060 OUT P+1,0
1070 J=0
1080 FOR I=0 TO 63
1090 OUT P,D
1100 J=J+1
```

```
1110 NEXT I
1120 D=D+64
1130 IF J=128 THEN RETURN ELSE 1080
```

Figure 17 - Color Bar Generator

4.3.4 Memory Test (See Figure 18)

```
10 REM THIS IS A MEMORY TEST PROGRAM FOR THE
20 REM CGS-800. THE ASSEMBLY LANGUAGE PART OF
30 REM THIS PROGRAM MUST BE LOADED INTO MEMORY
40 REM PRIOR TO RUNNING THIS PROGRAM.
50 REM THIS PART INITIALIZES THE ERROR CODES.
60 P=0 \ REM PASS NUMBER
70 REM YOU CAN STOP ON ERROR BY CHANGING
80 REM THE HALT FLAG - 0=HALT, 1=CONTINUE
90 FILL 260,255 \ REM HALT FLAG
100 FILL 256,0 \ REM LOW SCREEN ADDR
110 FILL 257,0 \ REM HIGH SCREEN ADDR
120 FILL 258,0 \ REM ERROR DATA
130 FILL 259,0 \ REM ERROR COUNT
140 C=CALL(0000) \ REM CALL MEMORY TEST
150 P=P+1 \ REM PASS NUMBER
160 E=EXAM(259)
170 A1=EXAM(256)
180 A2=EXAM(257)
190 D=EXAM(258)
200 A=A2*256+A1
210 PRINT
220 PRINT "PASS NUMBER ",P," COMPLETE"
230 PRINT "TOTAL NUMBER OF ERRORS= ",E
240 PRINT "LAST ERROR ADDRESS IN DECIMAL= ",A
250 PRINT "LAST ERROR DATA IN DECIMAL= ",D
260 GOTO 140
```

```
0000                    0010 *
0000                    0020 * COPYRIGHT BIOTECH ELECTRONICS
0000                    0030 * ALL RIGHT RESERVED
0000                    0040 *
0000                    0050 * PROGRAMMER: RICK STELLMACHER
0000                    0060 * SOURCE FILE: MTEST
0000                    0070 * VERSION: 2.0
0000                    0080 *
0000                    0090 * MTEST- MEMORY TEST DIAGNOSTICS
0000                    0100 * FOR THE CGS-800
0000                    0110 *
0000                    0120 * SYSTEM EQUATES
0000                    0130 PORT   EQU   128          * PORT ADDRESS
0000                    0140 SRMEM  EQU   6000H        * SCREEN ADDRESS
0000                    0150 EC     EQU   100H         * ERROR CODES
0000                    0160 ERRA   EQU   EC           * LAST ERROR ADDRESS
```

```
0000                    0170 ERRD   EQU  EC+2        * LAST ERROR DATA
0000                    0180 ERRC   EQU  EC+3        * ERROR COUNT
0000                    0190 HLTF   EQU  EC+4        * HALT FLAG
0000                    0200 CSA    EQU  EC+5        * 2 BYTES FOR SCREEN ADDR
0000                    0210 *
0000                    0220 *
0000                    0230 * MTEST- USES OP-CODE 5 AND 6 TO READ AND WRITE
0000                    0240 *       TO THE SCREEN ADDRESS MEMORY.  256 BYTES
0000                    0250 *       ARE TRANSFERED PER READ/WRITE SCREEN.
0000                    0260 *       THERE ARE 24 OPERATION PER SCREEN
0000                    0270 *       AND 256 PASSES.
0000                    0280 *
0000                    0290 *
0000                    0300 * INITIALIZE
0000                    0310 *
0000                    0320 MTEST  EQU  $
0000 21 00 60           0330        LXI  H,SRMEM     * SCREEN MEMORY ADDRESS
0003 22 05 01           0340        SHLD CSA         * SAVE SCREEN ADDRESS MEMORY
0006 1E 17              0350        MVI  E,23        *
0008 0E FF              0360        MVI  C,255       *
000A                    0370 *
000A                    0380 * FILL SCREEN
000A                    0390 *
000A CD 52 00           0400 NEXT   CALL WRSCN       * FILL 256 BYTES OF SCREEN
000D 16 FF              0410        MVI  D,255       * BYTE COUNTER
000F 23                 0420 LOOP1  INX  H           * THIS PART CALCUATES A DATA PATTERN
0010 7D                 0430        MOV  A,L         * TO BE PUT ON THE SCREEN
0011 84                 0440        ADD  H           * ACC= H+L
0012 81                 0450        ADD  C           * ACC=H+L+DATA PATTERN
0013 47                 0460        MOV  B,A         * PUT IN B BEFORE CALLING POUT
0014 CD 76 00           0470        CALL POUT        * PUT ON SCREEN
0017 15                 0480        DCR  D           * ONE LESS BYTE
0018 C2 0F 00           0490        JNZ  LOOP1       * TRANSFER MORE BYTES
001B                    0500 *
001B                    0510 * VERIFY SCREEN
001B                    0520 *
001B 2A 05 01           0530        LHLD CSA         *
001E CD 6D 00           0540        CALL RDSCN       * READ SCREEN
0021 CD 80 00           0550        CALL PIN         *
0024 16 FF              0560        MVI  D,255       * BYTE COUNTER
0026 23                 0570 LOOP2  INX  H           *
0027 CD 80 00           0580        CALL PIN         *
002A 94                 0590        SUB  H           * THIS PART CALCULATES WHAT THE
002B 91                 0600        SUB  C           * ORIGINAL DATA SHOULD BE.
002C BD                 0610        CMP  L           * SAME DATA?
002D C4 89 00           0620        CNZ  MERR        * NO
0030 15                 0630        DCR  D           *
0031 C2 26 00           0640        JNZ  LOOP2       * YES
0034                    0650 *
0034                    0660 * UPDATE POINTERS
0034                    0670 *
0034 24                 0680        INR  H           * ADD 256 TO H AND L
0035 2E 00              0690        MVI  L,00H       *
0037 22 05 01           0700        SHLD CSA         * SAVE CURRENT SCREEN ADDRESS
```

```
003A 7B             0710        MOV   A,E          *
003B FE 00          0720        CPI   00           * DONE?
003D CA 45 00       0730        JZ    CHPAT        * JUMP IF DONE TO CHANGE PATTERN
0040 3D             0740        DCR   A            *
0041 5F             0750        MOV   E,A          *
0042 C3 0A 00       0760        JMP   NEXT         *
0045                0770 CHPAT  EQU   $
0045 0D             0780        DCR   C            * DATA PATTERN-1
0046 1E 17          0790        MVI   E,23         *
0048 21 00 60       0800        LXI   H,SRMEM      * TOP OF SCREEN
004B 22 05 01       0810        SHLD  CSR          * SAVE CURRENT SCREEN ADDRESS
004E C2 0A 00       0820        JNZ   NEXT         * COMPLETELY DONE?
0051 C9             0830        RET                * 256 PASSES COMPLETE
0052                0840 *
0052                0850 * WRITE SCREEN SUBROUTINE
0052                0860 *
0052                0870 WRSCN  EQU   $
0052 06 06          0880        MVI   B,6          * OP-CODE 6 WRITE SCREEN
0054 CD 76 00       0890        CALL  POUT         *
0057 45             0900        MOV   B,L          * LOW SCREEN ADDRESS
0058 CD 76 00       0910        CALL  POUT         *
005B 44             0920        MOV   B,H          * HIGH SCREEN ADDRESS
005C CD 76 00       0930        CALL  POUT         *
005F 06 8F          0940        MVI   B,8FH        * HIGH DENSITY MODE
0061 CD 76 00       0950        CALL  POUT         *
0064 06 FF          0960        MVI   B,255        * NUMBER OF BYTES
0066 CD 76 00       0970        CALL  POUT         *
0069 CD 9C 00       0980        CALL  INIT         *
006C C9             0990        RET                *
006D                1000 *
006D                1010 * READ SCREEN SUBROUTINE
006D                1020 *
006D                1030 RDSCN  EQU   $
006D 06 05          1040        MVI   B,5          * OP-CODE 5 READ SCREEN
006F CD 76 00       1050        CALL  POUT         *
0072 CD 9C 00       1060        CALL  INIT         *
0075 C9             1070        RET                *
0076                1080 *
0076                1090 * OUTPUT DATA SUBROUTINE
0076                1100 *
0076                1110 POUT   EQU   $
0076 DB 81          1120        IN    PORT+1       * GET STATUS
0078 1F             1130        RAR                * BIT 0= STATUS
0079 DA 76 00       1140        JC    POUT         *
007C 78             1150        MOV   A,B          *
007D D3 80          1160        OUT   PORT         * PASS PARAMETER
007F C9             1170        RET                *
0080                1180 *
0080                1190 * INPUT DATA SUBROUTINE
0080                1200 *
0080                1210 PIN    EQU   $
0080 DB 81          1220        IN    PORT+1       * INPUT STATUS
0082 17             1230        RAL                * BIT 7
0083 DA 80 00       1240        JC    PIN          *
```

```
0086 DB 80              1250        IN   PORT        * GET BYTE
0088 C9                 1260        RET              *
0089                    1270 *
0089                    1280 * LOG ERROR ROUTINE
0089                    1290 *
0089                    1300 MERR   EQU  $
0089 22 00 01           1310        SHLD ERRA        * SAVE LAST ERROR ADDRESS
008C 32 02 01           1320        STA  ERRD        * SAVE LAST ERROR DATA
008F 3A 03 01           1330        LDA  ERRC        * GET ERROR COUNT
0092 3C                 1340        INR  A           *
0093 32 03 01           1350        STA  ERRC        * SAVE ERROR COUNT
0096 3A 04 01           1360        LDA  HLTF        * GET HALT FLAG
0099 A7                 1370        ANA  A           * TEST IF SET
009A C0                 1380        RNZ              * DONT STOP
009B 76                 1390        HLT              * STOP
009C                    1400 *
009C                    1410 * EXECUTE OP-CODE SUBROUTINE
009C                    1420 *
009C                    1430 INIT   EQU  $
009C DB 81              1440        IN   PORT+1      * GET STATUS
009E 1F                 1450        RAR              * TEST BIT 0
009F DA 9C 00           1460        JC   INIT        *
00A2 D3 81              1470        OUT  PORT+1      * INITIATE OP-CODE
00A4 C9                 1480        RET              * RETURN
00A5                    1490 *

SYMBOL TABLE

CHPAT 0045   CSR   0105   EC    0100   EPRA  0100   ERRC  0103
ERRD  0102   HLTF  0104   INIT  009C   LOOP1 000F   LOOP2 0026
MERR  0089   MTEST 0000   NEXT  000A   PIN   0080   PORT  0090
POUT  0076   RDSCN 006D   SRMEM 6000   WRSCN 0052
```

```
D 0000 00A4
0000 21 00 60 22 05 01 1E 17 0E FF CD 52 00 16 FF 23
0010 7D B4 81 47 CD 76 00 15 C2 0F 00 2A 05 01 CD 6D
0020 00 CD 80 00 16 FF 23 CD 80 00 94 91 BD C4 89 00
0030 15 C2 26 00 24 2E 00 22 05 01 7B FE 00 CA 45 00
0040 3D 5F C3 0A 00 0D 1E 17 21 00 60 22 05 01 C2 0A
0050 00 C9 06 06 CD 76 00 45 CD 76 00 44 CD 76 00 06
0060 8F CD 76 00 06 FF CD 76 00 CD 9C 00 C9 06 05 CD
0070 76 00 CD 9C 00 C9 DB 81 1F DA 76 00 78 D3 80 C9
0080 DB 81 17 DA 80 00 DB 80 C9 22 00 01 32 02 01 3A
0090 03 01 3C 32 03 01 3A 04 01 A7 C0 76 DB 81 1F DA
00A0 9C 00 D3 81 C9
```

Figure 18 - Memory Test

APPENDIX A

Firmware Pack I

```
0000                    0010 * COPYRIGHT BIOTECH ELECTRONICS  1979
0000                    0020 * ALL RIGHTS RESERVED
0000                    0030 *
0000                    0040 * PROGRAMMER: RICK STELLMACHER
0000                    0050 * SOURCE FILE: FIRM1
0000                    0060 * VERSION:3.0
0000                    0070 *
0000                    0080 * FIRM1 - GRAPHICS/ALPHANUMERIC DRIVER
0000                    0090 * FOR THE CGS-888.
0000                    0100 *
0000                    0110 *  OP-CODE          OPERATION
0000                    0120 *  -------          ---------
0000                    0130 *    0      CLEAR SCREEN
0000                    0140 *    1      SET MODE
0000                    0150 *    2      PLOT POINT
0000                    0160 *    3      DRAW LINE GIVEN TWO END POINTS
0000                    0170 *    4      ALPHANUMERICS/SEMIGRAPHICS DRVR
0000                    0180 *    5      READ SCREEN
0000                    0190 *    6      WRITE SCREEN
0000                    0200 *
0000                    0210 * THE FORMAT IS AN OP-CODE FOLLOWED BY THE
0000                    0220 * NECESSARY PARAMETERS THEN OUTPUT TO THE
0000                    0230 * INITIATE PORT.  THEN TEST THE STATUS PORT
0000                    0240 * FOR THE STATUS CODE.
0000                    0250 *
0000                    0260 *          STATUS CODE
0000                    0270 *          -----------
0000                    0280 * BIT  0   NOT USED
0000                    0290 *       1   INVALID OP-CODE
0000                    0300 *       2   RESET REQUIRED
0000                    0310 *       3   NOT USED
0000                    0320 *       4   NOT USED
0000                    0330 *       5   NOT USED
0000                    0340 *       6   INPUTTING PARAMETERS
0000                    0350 *       7   BUSY/DONE EXCUTING OP-CODE
0000                    0360 *
0000                    0370 *
0000                    0380 * SYSTEM EQUATES
0000                    0390 *
0000                    0400 *
0000                    0410 * I/O PORT EQUATES
0000                    0420 INPT   EQU   00H          * INPUT DATA PORT
0000                    0430 OUTPT  EQU   00H          * OUTPUT DATA PORT
0000                    0440 STPT   EQU   01H          * STATUS PORT
0000                    0450 OEPT   EQU   02H          * OUTPUT EXPANSION PORT
0000                    0460 IEPT   EQU   02H          * INPUT EXPANSION PORT
0000                    0470 MDPT   EQU   01H          * OUTPUT MODE PORT
0000                    0480 *
```

```
0000                    0490 * MEMORY EQUATES
0000                    0500 SRMEM  EQU   6000H       * SCREEN REFRESH MEMORY
0000                    0510 PBUFF  EQU   4000H       * PARAMETER BUFFER AREA
0000                    0520 PRAM   EQU   4030H       * START OF PARAMETER STORAGE
0000                    0530 STACK  EQU   40FFH       * TOP OF STACK- 30 BYTES
0000                    0540 DBASE  EQU   4100H       * BEGIN OF DATA BASE- 768 BYTES
0000                    0550 FWA2   EQU   200AH       * BEGINING OF FWP2 LOOK-UP TBL
0000                    0560 EXPTI  EQU   2006H       * INPUT EXPANSION PORT
0000                    0570 EXPTO  EQU   2003H       * OUTPUT EXPANSION PORT
0000                    0580 HIDEN  EQU   2009H       * HIDDEN DOT ROUTINE
0000                    0590 *
0000                    0600 * PARAMETER BUFFER EQUATES
0000                    0610 NOL    EQU   PBUFF       * NO. OF LINES
0000                    0620 BPL    EQU   PBUFF+1     * BYTES PER LINE
0000                    0630 CSMD   EQU   PBUFF+2     * CMD SEQ MODE
0000                    0640 THRED  EQU   PBUFF+3     * 3D MODE
0000                    0650 UPMOD  EQU   PBUFF+4     * UPDATE MODE
0000                    0660 MD     EQU   PBUFF+5     * MODE
0000                    0670 PCNT   EQU   PBUFF+6     * PARAMETER COUNT
0000                    0680 STAT   EQU   PBUFF+7     * STATUS
0000                    0690 Z      EQU   PBUFF+8     * COLOR/INTENSITY
0000                    0700 X0     EQU   PBUFF+9     * CURSOR X COORD
0000                    0710 Y0     EQU   PBUFF+10    * CURSOR Y COORD
0000                    0720 X      EQU   PBUFF+11    * CURRENT X COORD
0000                    0730 Y      EQU   PBUFF+12    * CURRENT Y COORD
0000                    0740 DX     EQU   PBUFF+13    * DELTA X
0000                    0750 DY     EQU   PBUFF+14    * DELTA Y
0000                    0760 SDX    EQU   PBUFF+15    * SPECIAL DELTA X
0000                    0770 SDY    EQU   PBUFF+16    * SPECIAL DELTA Y
0000                    0780 WEFLG  EQU   PBUFF+17    * WRITE/ERASE FLAG
0000                    0790 CNTR   EQU   PBUFF+18    * LOOP CNTR=NO. OF DOTS
0000                    0800 LONG   EQU   PBUFF+19    * LONGEST DELTA
0000                    0810 HAFL   EQU   PBUFF+21    * LONG/2
0000                    0820 LXIS   EQU   PBUFF+23    * ADDR OF LONG AXIS
0000                    0830 SXIS   EQU   PBUFF+25    * ADDR OF SHORT AXIS
0000                    0840 SHORT  EQU   PBUFF+27    * SHORTEST DELTA
0000                    0850 LCUR   EQU   PBUFF+29    * LAST CURSOR
0000                    0860 FLAG1  EQU   PBUFF+31    * USER DEFINED FLAG
0000                    0870 FLAG2  EQU   PBUFF+32    * USER DEFINED FLAG
0000                    0880 CHARA  EQU   PBUFF+33    * CHARACTER ADDRESS
0000                    0890 LINEA  EQU   PBUFF+34    * LINE ADDRESS
0000                    0900 *
0000                    0910 * START OF FIRMWARE PACK 1
0000                    0920 *
0000                    0930 BEGIN  ORG   0000H
0000                    0940 *
0000                    0950 * INITIALIZATION ROUTINE
0000                    0960 *
0000                    0970 * THE PROCESSOR EXECUTES THIS CODE ON POWER-ON
0000                    0980 * OR BOARD RESET.  THIS CODE WILL CLEAR THE SCREEN
0000                    0990 * SET THE MODE TO HIGH DENSITY AND INITIALIZE THE
```

```
0000                    1000 * PARAMETERS.
0000                    1010 *
0000 31 FF 40           1020        LXI   SP,STACK      * TOP OF STACK-30 BYTES
0003 21 31 40           1030        LXI   H,PRAM+1      * PARAMETER STORAGE
0006 AF                 1040        XRA   A             * CLEAR ACC
0007 32 03 40           1050        STA   THRED         * NOT 3D
000A 3C                 1060 .      INR   A             * A=01
000B 36 8F              1070        MVI   M,8FH         * HIGH DENSITY MODE
000D CD 04 00           1080        CALL  DRVR          * EXECUTE OP-CODE 01
0010 AF                 1090        XRA   A             * CLEAR ACC
0011 36 00              1100        MVI   M,00H         * CLEAR AND INITIALIZE
0013 CD 04 00           1110        CALL  DRVR          * EXECUTE OP-CODE 00
0016 C3 53 00           1120        JMP   IDLP          * GOTO MAIN IDLE LOOP
0019                    1130 *
0019                    1140 * RESET REQUIRED
0019                    1150 * THE PROCESSOR JUMPS TO THIS LOCATION WHEN
0019                    1160 * A HARD ERROR HAS OCCURED AND A BOARD RESET
0019                    1170 * MAY BE REQUIRED.
0019                    1180 *
0019                    1190 RSTR2  EQU   $
0019 06 04              1200        MVI   B,04H         * RESET REQUIRED STATUS
001B CD 22 00           1210        CALL  OSTAT         * OUTPUT STATUS
001E C3 53 00           1220        JMP   IDLP          * RETURN TO IDLE LOOP
0021 00                 1230        NOP                 *
0022                    1240 *
0022                    1250 * OUTPUT STATUS
0022                    1260 *
0022                    1270 * OUTPUT THE STATUS CODE IN THE B REGISTER
0022                    1280 *
0022                    1290 OSTAT  EQU   $
0022 3A 07 40           1300        LDA   STAT          * GET OLD STATUS
0025 B0                 1310        ORA   B             * DON'T CHANGE OTHER BITS
0026 32 07 40           1320        STA   STAT          * SAVE NEW STATUS
0029 D3 00              1330        OUT   OUTPT         * OUTPUT STATUS
002B C9                 1340        RET                 * RETURN TO MAIN ROUTINE
002C                    1350 *
002C                    1360 * EXPANSION PORT-OUTPUT RST 5.5 -2CH-
002C                    1370 *
002C                    1380 * WHEN DATA HAS BEEN TAKEN FROM THE EXPANSION
002C                    1390 * OUTPUT PORT -U10- THE PROCESSOR WILL VECTOR TO
002C                    1400 * THIS LOCATION -2CH-
002C                    1410 *
002C C3 03 20           1420 OEXPT  JMP   EXPTO         * JUMPS TO PWP2
002F                    1430 *
002F                    1440 *
002F                    1450 * OP-CODE ERROR ROUTINE
002F                    1460 *
002F                    1470 OPERR  EQU   $
002F 06 02              1480        MVI   B,02H         * OP-CODE ERROR
0031 C3 22 00           1490        JMP   OSTAT         * OUTPUT STATUS
0034                    1500 *
```

```
0034                    1510 * EXPANSION PORT-INPUT  RST 6.5 -34H-
0034                    1520 *
0034                    1530 * WHEN DATA HAS BEEN CLOCKED INTO THE INPUT
0034                    1540 * EXPANSION PORT -U9- THE PROCESSOR WILL VECTOR
0034                    1550 * TO THIS LOCATION -34H-
0034                    1560 *
0034 C3 06 20           1570 IEXPT  JMP   EXPTI         * JUMPS TO FWP2
0037 00                 1580        NOP
0038                    1590 *
0038                    1600 * PROCESSOR JUMPES TO THIS ADDRESS WHEN
0038                    1610 * A HARD ERROR OCCURS -RST 7.0  -38H-
0038                    1620 *
0038 C3 19 00           1630 RST7   JMP   RSTR0         * HARD ERROR RESET
003B 00                 1640        NOP
003C                    1650 *
003C                    1660 * INITIATE OP-CODE
003C                    1670 * THE PROCESSOR WILL JUMP THE ADDRESS -3CH- WHEN
003C                    1680 * THE HOST SYSTEM OUTPUTS ANY DATA TO THE PORT
003C                    1690 * ADDRESS+1.  ALL PARAMETERS MUST HAVE BEEN
003C                    1700 * PREVIOUSLY STORED IN THE PARAMETER STORAGE -PRAM-.
003C                    1710 * RST 7.5  -3CH-
003C                    1720 *
003C                    1730 INIT   EQU   $
003C FB                 1740        EI                  * ENABLE INTERRUPTS
003D 78                 1750        MOV   A,B           * PARAM CNT
003E 32 06 40           1760        STA   PCNT          * SAVE PARAMETER COUNT
0041 3E 00              1770        MVI   A,00H         * BUSY EXECUTING OP-CODE
0043 32 07 40           1780        STA   STAT          * SAVE NEW STATUS AND CLEAR
0046 D3 00              1790        OUT   OUTPT         * OUTPUT STATUS
0048 21 30 40           1800        LXI   H,PRAM        * INIT RAM POINTER
004B 7E                 1810        MOV   A,M           * GET OP-CODE
004C 23                 1820        INX   H             * FIRST PARAMETER
004D CD 84 00           1830        CALL  DRVR          * EXEC OP-CODE
0050 CD 22 00           1840        CALL  OSTAT         * OUTPUT STATUS
0053                    1850 *
0053                    1860 * MAIN IDLE LOOP
0053                    1870 * THIS ROUTINE SCANS THE S-100 BUS FOR NEW
0053                    1880 * OP-CODES AND PARAMETERS, AND ENTERS IT IN
0053                    1890 * THE PARAMETER STORAGE AREA.
0053                    1900 * THE PROCESSOR WILL REMAIN IN THE ILDE LOOP
0053                    1910 * UNTIL THE HOST PROCESSOP OUTPUTS ANY DATA
0053                    1920 * TO THE PORT ADDRESS+1.  THIS WILL INTERRUPT
0053                    1930 * THE PROCESSOR AND VECTOR TO LOCATION -3CH-
0053                    1940 *
0053                    1950 IDLP   EQU   $
0053 31 FF 40           1960        LXI   SP,STACK      * RE-INITIALIZE SP
0056 FB                 1970        EI                  * ENABLE INTERRUPTS
0057 3E 1B              1980        MVI   A,1BH         * MASK RST 6.5, RST 5.5
0059 00                 1990        NOP                 * SIM INSTRUCTION GOES HERE
005A 21 30 40           2000        LXI   H,PRAM        * INITAL PARAM STORAGE AREA
005D 06 40              2010        MVI   B,40H         * INPUTING PARAMETERS
```

```
005F CD 22 00           2020        CALL  OSTAT       * OUTPUT STATUS
0062 CD 70 00           2030        CALL  GPRAM       * GET OP-CODE PARAMETER
0065 23                 2040        INX   H            * NEXT PARAM POS
0066 06 00              2050        MVI   B,00H        * SET PRMCNT =0
0068 CD 70 00           2060 SLP2   CALL  GPRAM       * GET PARAMETER
006B 23                 2070        INX   H            * BUMP POINTER
006C 04                 2080        INR   B            * BUMP PARAM CNTR
006D C3 68 00           2090        JMP   SLP2         * GO GET MORE
0070                    2100 *
0070                    2110 *
0070                    2120 * GPRAM- GETS USER PARAMETER FROM INPUT PORT
0070                    2130 * ENTER- HL=ADDR TO STORE PARAM
0070                    2140 * DOES NOT RETURN UNTIL PARAMETER HAS BEEN
0070                    2150 * ENTERED INTO THE INPUT PORT.
0070                    2160 *
0070                    2170 GPRAM  EQU   $
0070 DB 01              2180        IN    STPT         * GET INPUT STATUS
0072 07                 2190        RLC                * 0=DATA AVAIL
0073 DA 70 00           2200        JC    GPRAM        * WAIT TIL DATA AVAIL
0076 DB 00              2210        IN    INPT         * GET DATA FROM INPUT PORT
0078 77                 2220        MOV   M,A          * SAVE DATA IN BUFFER
0079 C9                 2230        RET                *
007A                    2240 *
007A                    2250 *
007A                    2260 * PPRAM- SENDS DATA FROM MEMORY
007A                    2270 * TO OUTPUT PORT 00H.
007A                    2280 * ENTER- H,L REG = ADDR OF STORED DATA
007A                    2290 *
007A                    2300 PPRAM  EQU   $
007A DB 01              2310        IN    STPT         * GET STATUS
007C 0F                 2320        RRC                * CHECK DATA AVAIL
007D DA 7A 00           2330        JC    PPRAM        * WAIT TIL DATA AVAL
0080 7E                 2340        MOV   A,M          * GET BYTE TO OUTPUT
0081 D3 00              2350        OUT   OUTPT        * OUTPUT DATA
0083 C9                 2360        RET                *
0084                    2370 *
0084                    2380 *
0084                    2390 * LOOK-UP TABLE FOR OP-CODES
0084                    2400 * ENTER
0084                    2410 * A REG= OP-CODE
0084                    2420 * H,L REG= FIRST LOCATION OF PARAMETERS
0084                    2430 *
0084                    2440 * SAVES ALL REGISTERS EXCEPT THE A REG AND
0084                    2450 * THE B AND C REG'S.
0084                    2460 *
0084                    2470 * EXIT
0084                    2480 * D,E REG = FIRST LOCATION OF PARAMETERS
0084                    2490 * A REG = PARAMETER COUNT
0084                    2500 * PC = FIRST WORD ADDRESS OF OP-CODE
0084                    2510 *
0084                    2520 * THIS ROUTINE TAKES THE OP-CODE AND CALCULATES
```

Firmware Pack I (Con't.)

```
0084                    2530 * THE FIRST WORD ADDRESS OF THAT OP-CODE SUBROUTINE
0084                    2540 *
0084                    2550 DRVR   EQU   $
0084 FE 1E              2560        CPI   30            * MAXIMUM OP-CODE
0086 D2 2F 00           2570        JNC   OPERR         * OP-CODE ERROR
0089 D5                 2580        PUSH  D             *
008A E5                 2590        PUSH  H             *
008B EB                 2600        XCHG                * DE- PARAMETER POINTER
008C 21 A4 00           2610        LXI   H,JPTBL       * JUMP TABLE
008F FE 07              2620        CPI   7             * NO. OF OP-CODE + 1
0091 DA 97 00           2630        JC    CRT0          * IS IT IN FIRM2?
0094 21 FC 1F           2640        LXI   H,FWA2-14     * BEGINING OF ADDR TABLE
0097 07                 2650 CRT0   RLC                 *
0098 06 00              2660        MVI   B,00H         *
009A 4F                 2670        MOV   C,A           * C=OP-CODE
009B 09                 2680        DAD   B             *
009C 7E                 2690        MOV   A,M           * MOVE JMP TABLE ADDR TO HL
009D 23                 2700        INX   H
009E 66                 2710        MOV   H,M
009F 6F                 2720        MOV   L,A
00A0 3A 06 40           2730        LDA   PCNT          * ACC=PARAMETER COUNT
00A3 E9                 2740 CRT1   PCHL                * GO TO ROUTINE OF OP-CODE
00A4 EC 03              2750 JPTBL  DW    CLRSR         * CLEAR SCREEN
00A6 B7 00              2760        DW    SMODE         * SET MODE
00A8 7B 01              2770        DW    PLTPT         * PLOT POINT
00AA 76 02              2780        DW    DRWLN         * DRAW LINE
00AC 6C 03              2790        DW    ALSEM         * ALPHA&JUMPICS/SEMIGRAPHICS
00AE 51 03              2800        DW    RDSCN         * READ SCREEN
00B0 43 03              2810        DW    WRSCN         * WRITE SCREEN
00B2 E1                 2820 CRTX   POP   H
00B3 D1                 2830        POP   D
00B4 06 80              2840        MVI   B,80H         * DONE EXECUTING OP-CODE
00B6 C9                 2850        RET                 * RETURN
00B7                    2860 *
```

TABLE 3 – DETAILED DESCRIPTION OF VDG MODES

VDG PINS									COLOR			TV SCREEN		VDG DATA BUS	COMMENTS
$\overline{MS}$	$\overline{A/G}$	$\overline{A/S}$	$\overline{INT}$/EXT	GM2	GM1	GM0	CSS	INV	Character Color	Background	Border	Display Mode	Detail		
1	0	0	0	X	X	X	0	0	Green	Black	Black	32 Characters across 16 Characters down	8 dots, 12 dots, 7, 5	extra, ASCII code	The ALPHANUMERIC INTERNAL mode uses an internal character generator which contains the following five dot by seven dot characters: @ABCDEFGHIJ KLMNOPQRSTUVWXYZ[\] ↑ ← SP !"#$%&'()*+,-./0123456789:;< = >? The six bit ASCII code leaves two bits free and these may be externally connected to the mode pins ($\overline{A/G}$, $\overline{A/S}$, $\overline{INT}$/EXT, GM2, GM1, GM0, CSS or INV).
								1	Black	Green					
							1	0	Orange	Black					
								1	Black	Orange					
1	0	0	1	X	X	X	0	0	Green	Black	Black	32 Characters across 16 Characters down		one row of custom characters	The ALPHANUMERIC EXTERNAL mode uses an external character generator as well as a row counter. Thus, custom character fonts are graphic symbol sets with up to 256 different eight dot × 12 dot "characters" may be displayed
								1	Black	Green					
							1	0	Orange	Black					
								1	Black	Orange					
1	0	1	0	X	X	X	X	X	Lx C2 C1 C0 Color 0 X X X Black 1 0 0 0 Green 1 0 0 1 Yellow 1 0 1 0 Blue 1 0 1 1 Red 1 1 0 0 Buff 1 1 0 1 Cyan 1 1 1 0 Magenta 1 1 1 1 Orange		Black	64 Display elements across 32 Display elements down	L_3 L_2 / L_1 L_0 one element	C_2 C_1 C_0 L_3 L_2 L_1 L_0	The SEMIGRAPHICS FOUR mode uses an internal "course graphics" generator in which a rectangle (eight dots by twelve dots) is divided into four equal parts. The luminance of each part is determined by a corresponding bit on the VDG data bus. The color of illuminated parts is determined by three bits
1	0	1	1	X	X	X	0	X	Lx C1 C0 Color 0 X X Black 1 0 0 Green 1 0 1 Yellow 1 1 0 Blue 1 1 1 Red 0 X X Black 1 0 0 Buff 1 0 1 Cyan 1 1 0 Magenta 1 1 1 Orange		Black	64 Display elements across 48 Display elements down	L_5 L_4 / L_3 L_2 / L_1 L_0	C_1 C_0 L_5 L_4 L_3 L_2 L_1 L_0	The SEMIGRAPHIC SIX mode is similar to the SEMIGRAPHIC FOUR mode with the following differences: The eight dot by twelve dot rectangle is divided into six equal parts. Color is determined by the two remaining bits.
1	1	X	X	0	0	0	0	X	C1 C0 Color 0 0 Green 0 1 Yellow 1 0 Blue 1 1 Red 0 0 Buff 0 1 Cyan 1 0 Magenta 1 1 Orange		Green	64 Display elements across 64 Display elements down	E_3 E_2 E_1 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The GRAPHICS ONE C mode uses a maximum of 1024 bytes of display RAM in which one pair of bits specifies one picture element.
							1	X			Buff				
1	1	X	X	0	0	1	0	X	Lx 0 1	Color Black Green	Green	128 Display elements across 64 Display elements down	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The GRAPHICS ONE R mode uses a maximum of 1024 bytes of display RAM in which one bit specifies one picture element.
							1	X	0 1	Black Buff	Buff				
1	1	X	X	0	1	0	0	X	Same color as Graphics one C		Green	128 Display elements across 64 Display elements down	E_3 E_2 E_1 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The GRAPHICS TWO C mode uses a maximum of 2048 bytes of display RAM in which one pair of bits specifies one picture element.
							1	X			Buff				
1	1	X	X	0	1	1	0	X	Same color as Graphics one R		Green	128 Display elements across 96 Display elements down	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The GRAPHICS TWO R mode uses a maximum of 1536 bytes of display RAM in which one bit specifies one picture element
							1	X			Buff				
1	1	X	X	1	0	0	0	X	Same color as Graphics one C		Green	128 Display elements across 96 Display elements down	E_3 E_2 E_1 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The GRAPHICS THREE C mode uses a maximum of 3072 bytes of display RAM in which one pair of bytes specifies one picture element.
							1	X			Buff				
1	1	X	X	1	0	1	0	X	Same color as Graphics one R		Green	128 Display elements across 192 Display elements down	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The GRAPHICS THREE R mode uses a maximum of 3072 bytes of display RAM in which one bit specifies one picture element
							1	X			Buff				
1	1	X	X	1	1	0	0	X	Same color as Graphics one C		Green	128 Display elements across 192 Display elements down	E_3 E_2 E_1 E_0	C_1 C_0 C_1 C_0 C_1 C_0 C_1 C_0	The GRAPHICS SIX C mode uses a maximum of 6144 bytes of display RAM in which one pair of bits specifies one picture element.
							1	X			Buff				
1	1	X	X	1	1	1	0	X	Same color as Graphics one R		Green	256 Display elements across 192 Display elements down	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	L_7 L_6 L_5 L_4 L_3 L_2 L_1 L_0	The GRAPHICS SIX R mode uses a maximum of 6144 bytes of display RAM in which one bit specifies one picture element.
							1	X			Buff				